**NPC.gd**

extends StaticBody2D

#extends Node

enum QuestStatus { NOT\_STARTED, STARTED, COMPLETED } #все статусы "квеста"

var quest\_status = QuestStatus.NOT\_STARTED #статус квеста на данный момент

var dialogue\_state = 0 #статус диалога на данный момент

var necklace\_found = false #статус нахождения подвески

var dialoguePopup

var player

func \_ready():

dialoguePopup = get\_tree().root.get\_node("/root/World/CanvasLayer/DialoguePopUp")

player = get\_tree().root.get\_node("/root/World/YSort/Player")

func talk(answer = ""):

# NPC, анимация = "говорит"

$AnimatedSprite.play("talk")

dialoguePopup.npc = self

dialoguePopup.npc\_name = "Ryu"

# показываем "действующий" диалог

match quest\_status:

QuestStatus.NOT\_STARTED:

match dialogue\_state:

0:

# Обновляем ситуацию с диалогом

dialogue\_state = 1

# Показываем диалог

dialoguePopup.dialogue = "Hello adventurer! I lost my necklace, can you find it for me?"

dialoguePopup.answers = "[A] Yes [B] No"

dialoguePopup.open()

1:

match answer:

"A":

# Обновляем ситуацию с диалогом

dialogue\_state = 2

# Показываем диалог

dialoguePopup.dialogue = "Thank you!"

dialoguePopup.answers = "[A] Bye"

dialoguePopup.open()

"B":

# Обновляем ситуацию с диалогом

dialogue\_state = 3

# Показываем диалог

dialoguePopup.dialogue = "If you change your mind, you'll find me here."

dialoguePopup.answers = "[A] Bye"

dialoguePopup.open()

2:

# Обновляем ситуацию с диалогом

dialogue\_state = 0

quest\_status = QuestStatus.STARTED

# Закрываем диалог

dialoguePopup.close()

# Set NPC's animation to "idle"

$AnimatedSprite.play("idle")

3:

# Обновляем ситуацию с диалогом

dialogue\_state = 0

# Закрываем диалог

dialoguePopup.close()

# Set NPC's animation to "idle"

$AnimatedSprite.play("idle")

QuestStatus.STARTED:

match dialogue\_state:

0:

# Обновляем ситуацию с диалогом

dialogue\_state = 1

# Показываем диалог

dialoguePopup.dialogue = "Did you find my necklace?"

if necklace\_found:

dialoguePopup.answers = "[A] Yes [B] No"

else:

dialoguePopup.answers = "[A] No"

dialoguePopup.open()

1:

if necklace\_found and answer == "A":

# Обновляем ситуацию с диалогом

dialogue\_state = 2

# Показываем диалог

dialoguePopup.dialogue = "You're my hero! Thank you!"

dialoguePopup.answers = "[A] You're welcome"

dialoguePopup.open()

else:

# Обновляем ситуацию с диалогом

dialogue\_state = 3

# Показываем диалог

dialoguePopup.dialogue = "Please, find it!"

dialoguePopup.answers = "[A] I will!"

dialoguePopup.open()

2:

# Обновляем ситуацию с диалогом

dialogue\_state = 0

quest\_status = QuestStatus.COMPLETED

# Закрываем диалог

dialoguePopup.close()

# Set NPC's animation to "idle"

$AnimatedSprite.play("idle")

#yield(get\_tree().create\_timer(0.5), "timeout")

#тут по идее должна быть награда

3:

# Обновляем ситуацию с диалогом

dialogue\_state = 0

# Закрываем диалог

dialoguePopup.close()

# Set NPC's animation to "idle"

$AnimatedSprite.play("idle")

QuestStatus.COMPLETED: #если квест выполнен

match dialogue\_state:

0:

# Обновляем ситуацию с диалогом

dialogue\_state = 1

# Показываем диалог

dialoguePopup.dialogue = "Thanks again for your help!"

dialoguePopup.answers = "[A] Bye"

dialoguePopup.open()

1:

# Обновляем ситуацию с диалогом

dialogue\_state = 0

# Закрываем диалог

dialoguePopup.close()

# Set NPC's animation to "idle"

$AnimatedSprite.play("idle")

func \_on\_Hurtbox\_area\_entered(area):

talk()

**Player.gd**

extends KinematicBody2D

const BOOST = 500

const MAX\_SPEED = 80

const FRICTION = 500

enum {

MOVE,

ROLL,

ATTACK

} #массив с типами движения

var state = MOVE

var roll\_vector = Vector2.DOWN

var velocity = Vector2.ZERO #начальное положение персонажа по вектору 0,0

# velocity - изменение положения персонажа во времени

# boost - изменяет скорость со временем | постепенно увеличивает

# friction - то же, что и boost, но по окончании движения

onready var animationPlayer = $AnimationPlayer # доступ к "ребенку" AnimationPlayer у Player

onready var animationTree = $AnimationTree # доступ к "ребенку" AnimationTree у Player

onready var animationState = animationTree.get("parameters/playback") # параметр стоять/бежать

func \_ready():

animationTree.active = true #анимация будет неактивна до начала игры

#движение персонажа

func \_physics\_process(delta):

match state:

MOVE:

move\_state(delta)

ROLL:

roll\_state(delta)

ATTACK:

attack\_state(delta)

func move\_state(delta):

var input\_vector = Vector2.ZERO #задаем начальный вектор

input\_vector.x = Input.get\_action\_strength("ui\_right") - Input.get\_action\_strength("ui\_left") #настраиваем с помощью векторов движение влево/вправо

input\_vector.y = Input.get\_action\_strength("ui\_down") - Input.get\_action\_strength("ui\_up") #настраиваем с помощью векторов движение вверх/вниз

#стабилизируем скорость на диагоналях

input\_vector = input\_vector.normalized()

if input\_vector != Vector2.ZERO: #если мы двигаемся(т.к. начальное положние вектора изменилось)

roll\_vector = input\_vector

animationTree.set("parameters/Idle/blend\_position", input\_vector) #запоминает позицию для остановки после бега

animationTree.set("parameters/Run/blend\_position", input\_vector) #запоминает позицию для бега

animationTree.set("parameters/Attack/blend\_position", input\_vector) #запоминает позицию для атаки

animationTree.set("parameters/Roll/blend\_position", input\_vector) #запоминает позицию для переката

animationState.travel("Run")

velocity = velocity.move\_toward(input\_vector \* MAX\_SPEED, BOOST \* delta )

else: #мы стоим

animationState.travel("Idle")

velocity = velocity.move\_toward(Vector2.ZERO, FRICTION \* delta)

velocity = move\_and\_slide(velocity) # запоминаем и возвращаем положение до взаимодействия с объектами

# умножение на дельту всегда, когда есть что-либо, изменяющееся со временем

# либо что-то, связанное с частотой кадров

if Input.is\_action\_just\_pressed("roll"):

state = ROLL

if Input.is\_action\_just\_pressed("attack"):

state = ATTACK

func roll\_state(delta): #функция для переворота

velocity = Vector2.ZERO

animationState.travel("Roll")

func attack\_state(delta): #функция для атаки

velocity = Vector2.ZERO

animationState.travel("Attack")

func attack\_animation\_finished(): #функция для включения в анимацию для остановки движения

state = MOVE

func roll\_animation\_finished(): #функция для включения в анимацию для остановки переворота

state = MOVE

#var target = $RayCast2D.get\_collider()

#func \_input(event):

func input():

if Input.is\_action\_pressed("attack"):

var target = $RayCast2D.get\_collider()

#var now = OS.get\_ticks\_msec()

if target.is\_in\_group("NPCs"):

target.talk()

return

Necklace.gd

tool

#extends Node

extends Area2D

var NPC

func \_ready():

NPC = get\_tree().root.get\_node("/root/WorldYSort/NPC")

#если игрок нашел ожерелье, то necklace найден

func \_on\_Necklace\_body\_entered(body):

if body.name == "Player":

get\_tree().queue\_delete(self)

NPC.necklace\_found = true

**GrassEffect.gd**

extends Node2D

onready var animatedSprite = $AnimatedSprite

func \_ready(): #проигрывание анимации

animatedSprite.frame = 0 #если не сделать этого, то анимация будет проигрываться с конца

animatedSprite.play("Animate")

func \_on\_AnimatedSprite\_animation\_finished():

queue\_free() #разрушаем анимацию после использования

**Grass.gd**

extends Node2D

func create\_grass\_effect():

var GrassEffect = load("res://Effects/GrassEffect.tscn") #доступ к эффекту

var grassEffect = GrassEffect.instance()

var world = get\_tree().current\_scene #добавлем сцену эффекта травы к корневой сцене

world.add\_child(grassEffect) #делаем добавленную сцену ребенком корневой сцены

grassEffect.global\_position = global\_position #настраиваем позицию эффекта травы

func \_on\_Hurtbox\_area\_entered(area):

create\_grass\_effect()

queue\_free()

**DialoguePopup.gd**

extends Popup

var npc\_name setget name\_set

var dialogue setget dialogue\_set

var answers setget answers\_set

func name\_set(new\_value):

npc\_name = new\_value

$ColorRect/NPCName.text = new\_value

func dialogue\_set(new\_value):

dialogue = new\_value

$ColorRect/Dialogue.text = new\_value

func answers\_set(new\_value):

answers = new\_value

$ColorRect/Answers.text = new\_value

func open(): #когда функция началась, игра останавливается

get\_tree().paused = true

popup() #функция вызывает функцию popup() и показывает ее диалог

$AnimationPlayer.playback\_speed = 60.0 / dialogue.length()

$AnimationPlayer.play("ShowDialogue") #вызывает функцию ShowDialog()

func close(): #функция закрытия диалога

get\_tree().paused = false #возобновляет игру

hide() #прячет диалог

func \_ready(): #если диалог закрыт, то не получает входных данных

set\_process\_input(false)

func \_on\_AnimationPlayer\_animation\_finished(anim\_name):

set\_process\_input(true)

var npc

func \_input(event): #функция, обрабатывающая входную информацию от игрока

if event is InputEventKey:

if event.scancode == KEY\_A: #если нажата клавиша А

set\_process\_input(false)

npc.talk("A") #отправляет NPC ответ через фукнцию talk()

elif event.scancode == KEY\_B: #если нажата клавиша B

set\_process\_input(false)

npc.talk("B") #отправляет NPC ответ через фукнцию talk()